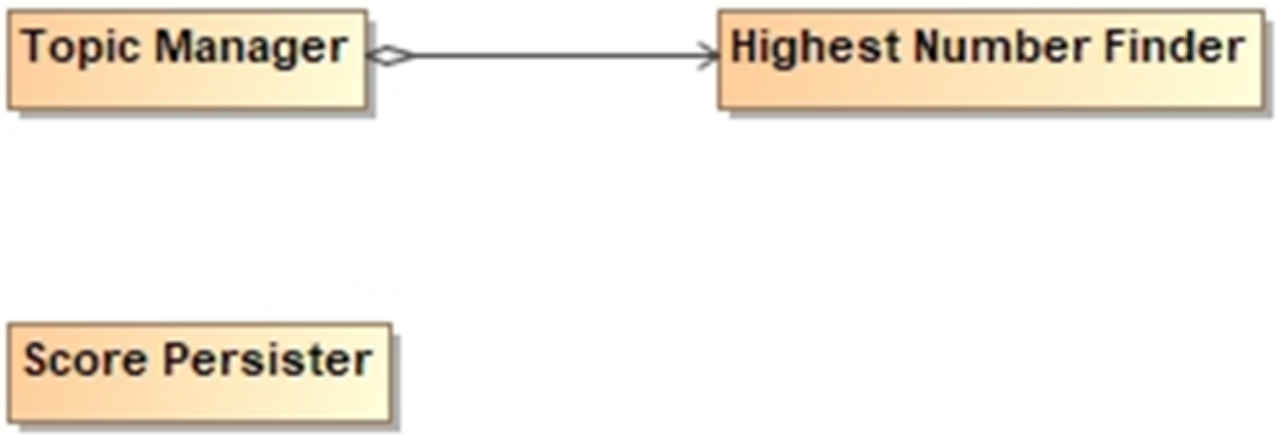
QLC-1) Find the Highest Number.

An organisation delivers several topics (subjects). Students are graded against each topic. You are required to store the top score for each topic.

We've designed the application so that it comprises of three core classes:

- A class to find the highest number from an array of integers.
- A class to find the highest score for a topic.
- A class to write the topic and score to a file on the disk.



You are going to follow a TDD approach to finding the highest number in an array of integers

Given the following specification

- If the input were {4, 5, -8, 3, 11, -21, 6} the result should be 11
- An empty array should throw an exception
- A single-item array should return the single item
- If several numbers are equal and highest, only one should be returned
- If the input were {7, 13} then the result should be 13
- If the input were {13, 4} then the result should be 13

Steps

1. Begin by creating a new maven project called `HighesNumberServices`
2. Create a package in the `test` folder called `com.demos.findhighestnumber`
3. Create a test class called to `HighestNumberFinderTests`
4. The most challenging part is determining which test to write first. Always start simple and with a test that will not need to handle exceptions.
   - So the simplest test we could do here is
     - A single-item array should return the single item
     - Write this first test, let's call the test method array_of_one_item_returns_this_item()

```
• package com.s2s.demos.findhighestnumber.v1;

  import org.junit.Test;
  import static org.junit.Assert.*;

  /**
   *
   * @author selvy
   */
  public class HighestNumberFinderTest
  {
      // TODO add test methods here.
      // The methods must be annotated with annotation @Test. For
  example:
      //
      @Test
      public void find_highest_in_array_of_one_expect_single_item()

      {
          // Arrange
          int array[] =
          {
              10
          };
          HighestNumberFinder cut = new HighestNumberFinder();
          int expectedResult = 10;

          // Act
          int result = cut.findHighestNumber( array );

          // Assert
          assertEquals( expectedResult, result );
      }
  }
```

5. Depending on the IDE you are using, most will allow you to create a class or method that doesn't currently exist. In the Test file HighestNumberFinderTests, try right-clicking on the HighestNumberFinder declaration and and see if there is an option to create the missing class. Make sure to create the class in the **src** folder and not the **test** folder

6. Try the same technique to create the method

```
        };
        HighestNumberFinder cut = new HighestNumberFinder();
        int expectedResult = 10;

        // Act
        int result = cut.findHighestNumber( array );
```

Create method "findHighestNumber(int[])" in com.s2s.demos.findhighestnumber.v1.HighestNumberFinder
Split into declaration and assignment

```
        assertEquals( expectedResult, result );
    }
```

7. Following a TDD approach, we get the IDE generate the production methods that match the tests
8. Now begin to work on the production code
   - Write enough production code to pass the test.
     - Do not be tempted to try and answer other parts of the requirements. Focus only on this requirement “A single item array should return the single item”

```
• class HighestNumberFinder
  {
      int findHighestNumber(int[] array)
      {
          return array[0];
      }
  }
```

     - Make sure the test passes
     - **A Golden rule of TDD -** if this was the only requirement then you have completed your task. Only write enough code to pass the test.
     - Commit your passing code to your git repo (never commit broken code)
9. Select the next requirement
   - I would suggest this one - If the input were {13, 4} then the result should be 13
     - Write the second test, let’s call the test method array_of_two_descending_items_return_first_item()

```
•     @Test
      public  void
  find_highest_in_array_of_two_descending_expect_first_element()
      {
          // Arrange
          int array[] = {13, 4};
          int expectedResult = 13;
          HighestNumberFinder cut = new HighestNumberFinder();

          // Act
          int result = cut.findHighestNumber(array);

          // Assert
          assertEquals(expectedResult, result);
      }
```

     - Write enough production code to pass the test. In this edge case, the production code does not change
     - Make sure the test passes

    - Commit your passing code to your git repo (never commit broken code)
10. Select the next requirement
    - I would suggest - If the input were {7, 13} then the result should be 13
        - Write the third test, let's call the test method array_of_two_ascending_items_return_last_item()

```
•         [Test]
          public void
  Array_of_two_ascending_items_return_last_item()
          {
              // Arrange
              int[] values = { 7, 13 };
              int expectedResult = 13;
              HighestNumberFinder cut = new HighestNumberFinder();

              // Act
              int result = cut.findHighestNumber(values);

              //Assert
              Assert.That(result, Is.EqualTo(expectedResult));
          }
```

        - Write enough production code to pass the test. Do not be tempted to try and answer the parts of the requirements. Focus only on this requirement "If the input were {7, 13} then the result should be 13"

```
•     int findHighestNumber(int[] array)
      {
          int highestSoFar = array[0];

          if( array.length > 1 && array[1] > highestSoFar )
              highestSoFar = array[1];

          return highestSoFar;
      }
```

    - Make sure the test passes
    - Commit your passing code to your git repo (never commit broken code)
    - The Production code has changed. Does it need to be refactored?
        - if yes, refactor the code
    - Make sure the test still passes
    - Commit your passing code to your git repo (never commit broken code)
11. Select each requirement and implement the test first then the production code

The steps above are known as RED, GREEN, REFACTOR

Git repo for solution